VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Mikko Niemikorpi

# Projektityökalun toteutus Pythonilla ja Djangolla

Liiketalous ja matkailu

2010

VAASAN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma

## TIIVISTELMÄ

| | |
|---|---|
| Tekijä | Mikko Niemikorpi |
| Opinnäytetyön nimi | Projektityökalun toteutus Pythonilla ja Djangolla |
| Vuosi | 2010 |
| Kieli | suomi |
| Sivumäärä | 30 |
| Ohjaaja | Sirkka Hellman |

---

Tietojenkäsittelyn koulutusohjelman opinnäytetyöni käsittelee suunnittelemaani ja toteuttamaani projektinhallinnan tueksi tehtyä www-sovellusta. Työ on toteutettu Python-ohjelmointikielellä ja Django-ohjelmistokehyksellä. Asetin työn tavoitteiksi oman projektinhallintaohjelmiston kehittämisen ja käytännön kokemuksen hankkimisen Django-kehyksestä. Idea työhön tuli omista tarpeista, kun en löytänyt opiskeluni aikana vastaan tulleiden projektien hallintaan sopivaa työkalua. Olemassa oleva tarve, kiinnostukseni www-sovellusten kehitykseen yleisesti ja kiinnostukseni Djangoon erityisesti saivat minut valitsemaan nykyisen aiheeni.

Käytännön osuudessa suunnittelin ja toteutin projektinhallinnan tueksi ohjelmiston, jossa on projektien ylläpito, tehtävien ylläpito, viestialue ja tiedostonhallinta. Teoriaa varten luin Mark Lutzin Programming Python -kirjaa, Ilkka Haikalan ja Jukka Märijärven kirjoittaman Ohjelmistotuotanto -kirjasta projektinhallinnasta, luin työtäni tukevia nettisivuja sekä katsoin muutaman Django-ohjelmistokehystä käsittelevän esityksen videomuodossa YouTubesta.

Työn lopputuloksena sain projektityökalun, jossa on suurin osa suunnittelemistani ominaisuuksista ja hyvät jatkokehitysmahdollisuudet sekä arvokasta käytännön kokemusta www-sovelluksien kehittämisestä ja Djangosta.

---

Avainsanat internet, ohjelmointi, projektinhallinta, projektit, Python, tietokantaohjelmat, verkko-ohjelmointi

VAASAN AMMATTIKORKEAKOULU

UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma

**ABSTRACT**

| | |
|---|---|
| Author | Mikko Niemikorpi |
| Title | Development of Project Management Tool with Python and Django |
| Year | 2010 |
| Language | Finnish |
| Pages | 30 |
| Name of Supervisor | Sirkka Hellman |

---

As my thesis work I designed and implemented a project management tool as a web application using Python programming language and Django framework. My goals in this work were to develop a tool for myself to help me managing of my projects and to get practical experience working with Django framework. I got the idea for the work during my studies when I needed a project management software for projects in classes but couldn't find suitable ones. A pre-existing need, my general interest in web applications and specific interest in Django led me to choose the topic.

For the practical part of the thesis work I designed and implemented a project management tool which includes tools to maintain projects and tasks, disscussion forum, and file storage. For the theoretical study I read "Programming Python" by Mark Lutz and selected parts about project management form "Ohjelmistotuotanto" by Ilkka Haikala and Jukka Märijärvi. I also read related websites and watched some presentations in video format from YouTube.

The results of my thesis work were a project management tool with most of the originally designed features, extensibility and valuable practical experience about developing web applications using Django framework.

---

Keywords Internet, Programming, Project Management, Projects, Python, Database Applications, Internet Programming

# SISÄLLYS

# 1 JOHDANTO

Sekä koulussa että vapaa-ajalla on vastaan tullut useita erilaisia projekteja. Projekteissani olisin halunnut käyttää hyödyksi tehokkaampia ja selkeämpiä työkaluja, mutta en löytänyt sopivia.
Tutkimani projektinhallintajärjestelmät olivat joko liian monimutkaisia tai suuria muutaman viikon projekteihin niiden jäsenmäärään verrattuna. En löytänyt yhtään sellaista, joka olisi täyttänyt kaikki toiveeni, joten päädyin jättämään projektinhallinnan perinteisten työkalujen, kuten sähköpostin ja tekstitiedostojen varaan.
Ohjelmointi ja erityisesti www-sovellusten ohjelmointi on kiinnostanut minua pitkään ja kun löysin alkuvuodesta 2009 Python-ohjelmointikielle tarkoitetun Django-nimisen ohjelmistokehyksen, idea omasta projektinhallintajärjestelmästä syntyi. Testiprojektien perusteella Django osoittautui helpoksi ja tehokkaaksi työkaluksi www-sovellusten toteuttamiseen ja päätin toteuttaa opinnäytetyökseni oman projektinhallintajärjestelmän.

## 1.1 Tausta

Opiskelen Vaasan ammattikorkeakoulussa neljättä vuotta ja päätin toteuttaa opinnäytetyökseni web-selaimella toimivan projektityökalun Python-ohjelmointikielellä ja sille rakennetulla Django-ohjelmistokehyksellä. Aloitin työn suunnittelun ja toteutuksen syksyllä 2009, mutta työn valmistuminen venyi keväälle 2010.
Valitsemani aihe kiinnostaa minua, koska koulun aikana on tullut vastaan erilaisia noin viikosta kuukauteen kestäviä projekteja, joiden hallintaan en löytänyt sopivaa työkalua. Päätin ohjelmoida oman ja www-sovellusten ohjelmointi on kiinnostanut minua jo pitkään, joten oman projektityökalun toteuttaminen opinnäytetyöksi oli varsin luonnollinen valinta.
Teoriaa varten lainasin muutaman Python-ohjelmointikieltä käsittelevän kirjan, luin Ilkka Haikalan ja Jukka Märijärven kirjoittaman Ohjelmistotuotanto -kirjasta projektinhallinnasta, luin työtäni käsitteleviä nettisivuja sekä katsoin muutaman Django-ohjelmistokehystä käsittelevän esityksen videomuodossa YouTubesta.

## 1.2 Tavoitteet ja rajaukset

Työn tavoitteena on toteuttaa selkeä ja helppokäyttöinen projektityökalu, joka on suunnattu lähinnä opiskelijoille muutamasta viikosta kuukauteen kestävien projektien läpiviennin tueksi. Halusin samalla saada käytännön kokemusta Djangolla toteutettavista isommista projekteista, koska olin ennen opinnäytetyötäni tehnyt sillä vain pienemmän mittakaavan testejä.
Rajasin työni ulkopuolelle projektinhallintaohjelmistoista yleensä löytyvät resurssien hallinnan ja seurannan sekä monimutkaisempia ominaisuuksia, kuten projektin tehtävien keskinäiset riippuvuudet ja projektin tunnuslukujen analysoinnin. Käyttämieni työkalujen modulaarisen ajattelun ansiosta nämä voidaan kuitenkin lisätä järjestelmään jälkeenpäin, jos tarvetta ja kiinnostusta ilmenee.

## 1.3 Työn rakenne

Luvussa kaksi käsittelen projektia ja projektinhallintaa yleisesti sekä käyn läpi muutamia tutkimiani ohjelmia.
Luvussa kolme esittelen käyttämäni Python-ohjelmointikielen, sen historiaa ja sen perustavia ideoita.
Kuvussa neljä käyn läpi Pythonilla toteutettua Django-ohjelmistokehystä, joka on suunniteltu www-sovellusten nopeaan toteuttamiseen. Esittelen kehyksen historiaa, sen suunnitteluun vaikuttaneita ideoita ja sen käyttökohteita.
Luvussa viisi esittelen oman työni suunnitelmat ja toteutuksen sekä kerron mietteitäni mahdollisesta jatkokehittämisestä.
Luvussa kuusi teen yhteenvedon opinnäytetyöstäni, jossa kerron kokemuksistani ja ajatuksistani sen aikana.

# 2 PROJEKTINHALLINTA

## 2.1 Yleistä

Projekti on yleensä kertaluontoinen tehtävä, jolle on määrätty resurssit, organisaatio ja joka toteutetaan ennalta laaditun aikataulun mukaan. Mikäli aikataulu puuttuu, on kyseessä joko kehityshanke tai ylläpitoprojekti, joita ei lasketa perinteisiksi projekteiksi. Projektille määrätyt resurssit voivat olla joko rahaa, henkilöitä, tiloja, laitteistoja tai aikaa. (Haikala & Märijärvi 2006, 225)

Projektinhallinnalla tarkoitetaan projektin suunnittelua ja seurantaa, jolla projekti saadaan vietyä läpi annetuilla resursseilla aikataulun puitteissa. Projektinhallinnan työkaluja ovat projektisuunnitelma, raportointi, työajan seuranta ja riskien hallinta. Projekti voidaan jakaa pienempiin osiin, jolloin sen seuranta helpottuu. Isompia osia kutsutaan aktiviteeteiksi ja aktiviteettien osia tehtäviksi. Tehtävä on yleensä muutaman viikon mittainen työkokonaisuus, jonka yksi henkilö pystyy hoitamaan. Aktiviteettien väleissä olevia tarkastuspisteitä kutsutaan etapeiksi, jolloin kyseisen aktiviteetin huolehtima välituote tai tavoite pitäisi olla valmiina. (Haikala & Märijärvi 2006, 225-228, 230-232.)

Projektihallintaohjelmistoa voidaan käyttää projektinhallinnan apuna. Sen avulla voidaan laskea projektin aikataulu, kun tiedetään projektin aktiviteetit ja tehtävät sekä näiden väliset riippuvuudet. Ohjelmistojen avulla projektin jäsenet voivat myös seurata resurssien käyttöä ja osoittaa niitä eri tehtävien täyttämiseen.

## 2.2 Valmiit ohjelmistot

**Microsoft Project** on Microsoftin projektinhallintajärjestelmä. Työpöytäsovelluksen käyttöliittymä on hyvin pitkälle rakennettu projektin alla olevien tehtävien ympärille. MS Project tallentaa projekti yhteen tiedostoon levylle. Microsoft Projectissa on resurssien ylläpito, tehtävien ylläpito, tehtävien väliset riippuvuudet, riippuvuuksien kuvaus esimerkiksi Gantt-kaaviolla ja kustannusten seuranta. (Microsoft, 2006; Microsoft, 2010a; Microsoft, 2010b.)

En käyttänyt MS Projectia omissa projekteissa koulun aikana, koska se tuntui työpöytäsovelluksena liian rajoittuneelta käytettäväksi usean luokan ja oppilaan välillä. Ominaisuuksiltaan ja käytettävyydeltään Microsoft Project olisi muuten ollut hyvä valinta.

**dotProject** on PHP:llä toteutettu web-pohjainen opensource-projektinhallintajärjestelmä, joka käyttää MySQL:ää tietokantanaan (dotProject-yhteisö, 2006b). Se oli alun perin vuonna 2000 dotMarketing Inc:n luoma avoin vaihtoehto Microsoft Projectille. Vuonna 2001 projekti siirtyi dotMarketingin kehityssivustolta Sourceforge.net-sivustolle ja sai vuonna 2002 uudet ylläpitäjät, kun edellinen ylläpitäjä ei enää ollut tavoitettavissa. Nykyään projekti on kokonaan vapaaehtoisten kehittäjien ylläpitämä ja kehittämä. (dotProject-yhteisö, 2006a.)

dotProject tarjoaa laajan valikoiman erilaisia työkaluja projektinhallinnan tueksi ja tukee useampaa projektia samassa järjestelmässä. Vakiomoduulien avulla voit ylläpitää projekteja, tehtäviä ja resursseja, ylläpitää yhtiöitä ja osastoja niiden sisällä, ylläpitää kalenterimerkintöjä ja luoda raportteja sekä kaavioita. Ulkopuolisten kehittäjien lisämoduuleilla projektiin saa helpdeskin tueksi suunnitellun järjestelmän ja laskutuksen. (dotProject-yhteisö, 2006c.)

Tutkin dotProjectin asennusta ja ylläpitoa opiskelujeni toisen vuoden alussa, mutta se oli aikaisemmista hyvistä kokemuksista huolimatta liian iso koulussa eteen tulleiden projektien hallinnointiin. dotProject on suunnattu yritysten ja yhteisöjen projektien hallintaan eikä sitä saa kovin helposti muokattua pienemmän käyttäjäryhmän tarpeisiin.

**Redmine** on Jean-Philippe Langin vuonna 2006 julkaistu projektinhallintajärjestelmä, joka on ohjelmoitu Ruby-ohjelmointikielellä ja Ruby on Rails -ohjelmistokehyksellä (Redmine-yhteisö, 2007a).). Redmineä levitetään GPL v2 -opensource-lisenssillä. Se on ottanut vaikutteita Trac-virheenhallintajärjestelmästä ja sisältää joitain samoja ominaisuuksia, kuten virheraporttien hallinta ja liityminen ulkopuolisiin versionhallintajärjestelmiin. Redmineä käyttävät mm. Ruby-ohjelmontikielen kehittäjät, Lighttpd-palvelimen kehittäjät ja Ryzom-nettiroolipelin

kehittäjät (Redmine-yhteisö, 2008.).

Redminen avulla voi ylläpitää useaa projektia, niiden tehtäviä ja ajankäyttö. Jokaiselle projektille luodaan oma wiki, keskustelualue, uutisjärjestelmä ja ne voidaan liittää johonkin ulkoiseen versionhallintajärjestelmään. (Redmine-yhteisö, 2007b.)

Redmine vaikuttaa juuri sopivan kokoiselta, mutta kuulin siitä vasta liian myöhään. Sain tietää Redminestä vasta aloitettuani oman järjestelmäni ohjelmoinnin ja kerrottuani siitä muutamalle tuttavalle.

Nimellä "mozrat" kulkeva käyttäjä on aloittanut projektinhallintaohjelmiston toteuttamisen Djangolle lokakuussa 2009. "**django-project-management**"-nimisestä projektista ei ole julkaistu vielä yhtään tiedostoja, joten en ole voinut tutustua järjestelmään tarkemmin. Kehittäjän lupaamien ominaisuuslistojen perusteella järjestelmästä on tulossa omaa toteutustani monipuolisempi ja vaikuttaa ainoalta aktiiviselta Djangolle toteutettavalta projektinhallintajärjestelmältä. (mozrat, 2009.)

# 3 PYTHON

## 3.1 Yleistä Pythonista

Python on tehokas ja selkeä korkean tason ohjelmointikieli, joka tarjoaa laajan moduulikirjaston erilaisten ongelmien ratkaisuun ja monipuoliset mahdollisuudet ajaa Python-koodia erilaisissa ympäristöissä. Kieli on oliopohjainen, mutta tukee myös perinteisempiä ohjelmointitapoja, kuten funktionaalista ja proseduraalista ohjelmointia. Pythonia ajetaan yleensä tulkattuna, joten nopeus saattaa muodostua ongelmaksi joissain tapauksissa. Ongelmaa voidaan kiertää käyttämällä Python-kääntäjiä ja siirtämällä ohjelman nopeuskriittiset osat esimerkiksi C-kielellä toteutettuihin moduuleihin. (Lutz 2006, foreword xiii-xx)

Kielen kehityksen on aloittanut Guido van Rossum vuonna 1989. Ensimmäinen julkinen versio kielestä oli versio 0.9, joka julkaistiin 1991. Tässä versiossa oli mukana kielelle ominaiset olio-ominaisuudet, poikkeuskäsittely ja moduulijärjestelmä. Versio 1.0 julkaistiin vuonna 1994 ja 2.0 vuonna 2000. Vuonna 2008 Pythonista julkaistiin taaksepäin yhteensopimaton versio 3.0, joka korjasi kieleen jääneitä ongelmia ja vanhoja toimintatapoja. Versiosarjat 2 ja 3 on suunniteltu elämään rinnakkain yhteensopivuussyistä vielä jonkin aikaa. Nykyään kieltä kehittää kielen käyttäjäyhteisö Python Software Foundationin ja Van Rossumin johdolla. (Lutz, 2006, 8-9)

Python Software Foundation on voittoa tavoittelematon organisaatio, joka perustettiin vuonna 2001. Sen tavoitteena on edistää ja suojata Pythonin kehitystä. Lisäksi se pitää hallussaan Pythoniin liittyviä tavaramerkkejä, järjestää ja tukee PyCon-tapahtumia sekä tarjoaa apurahoja valituille Pythoniin liittyville projekteille. (Python Software Foundation, 2002.)

Joulukuusta 2005 kielen kehityksen aloittanut Guido van Rossum on ollut Googlen palveluksessa. Hän työskentelee puolet työajastaan Pythonin kehityksen parissa ja loput ajasta Googlen projekteissa. Google käyttää sisäisesti paljon Pythonia, joten kielen kehittäjän palkkaaminen oli heille tärkeä investointi. (Van Rossum, 2008; Harrison, 2006.)

### 3.2 Kielen erikoispiirteitä

Pythonin suunnittelussa on käytetty seuraavia ideoita: ohjelmiston laatu, kehittäjän tuottavuus, ohjelman siirrettävyys ja integraatio.

Ohjelmiston laadulla viitataan Pythonin tapauksessa kielen tapaan kannustaa ja osittain jopa pakottaa selkeän ja helppolukuisen ohjelmakoodin kirjoittamiseen. Tämä saavutetaan mm. englanninkielisellä avainsanakäytännöllä välimerkkien sijaan ja sisennysten käyttöön ohjelmalohkojen erotteluun. Selkeys esiintyy Pythonissa myös pelkkää syntaksia syvemmällä.

Python kannustaa ohjelman modulaariseen rakentamiseen ja noudattaa itsekin tätä linjaa. Itse kielen ydin on hyvin pieni ja selkeä ja sen monipuolisuus saavutetaan lisäkirjastoilla ja -moduuleilla. (Lutz, 2006, 4)

Kehittäjän tuottavuus tarkoittaa, että Python huolehtii monesta kehittäjiä muissa kielissä hidastavista asioista. Esimerkiksi muuttujien tyyppimääritykset ja muistinhallinta on piilotettuna kielestä.

Selkeän kehittämisen ansiosta Python-ohjelmakoodit ovat paljon perinteisiä kieliä lyhyempiä, joka johtaa nopeampaan ohjelmistokehitykseen, pienempiin virhemääriin ja vähempään määrään ylläpidettävää koodia. Näiden ominaisuuksia ansiosta Pythonia käytetään nykyään muutaman rivin apuohjelmista tuhansien rivien tietojärjestelmiin. (Lutz, 2006, 4-5)

Ohjelman siirrettävyys viittaa Pythonin tarjoamiin mahdollisuuksiin ajaa sitä lähes missä tahansa ympäristössä ilman muokkauksia. Windowsin, Linuxin ja Mac OS:n lisäksi esimerkiksi joihinkin Nokian puhelimiin saa Python-ajoympäristön. Jokaisessa ympäristössä toimii lähes kaikki Pythonin peruskirjastot. Työpöytäympäristöihin kieli tarjoaa yhtenäisen käyttöliittymäkirjaston. (Lutz, 2006, 5)

Integraatiolla tarkoitetaan Pythonin avoimuutta toimia myös muiden järjestelmien kanssa. Pythonin pystyy sisällyttämään omaan ohjelmaan komentokieleksi ja sen avulla voidaan käyttää myös muiden ohjelmointikielien kirjastoja. Yksi esimerkki tästä on IronPython, joka mahdollistaa ohjelmoinnin Microsoftin .Net-alustalla ja Jython, joka on vastaavanlainen järjestelmä Sun Microsystemsin Javalle. Integraatio

muiden järjestelmien ja kirjastojen kanssa mahdollistaa Pythonin käytön eräänlaisena liimana näiden välillä ilman, että kehittäjä joutuu opettelemaan useaa eri ohjelmointikieltä ja suunnittelemaan omaa keskusteluprotokollaa järjestelmien välille. (Lutz, 2006, 5-6)

# 4 DJANGO

## 4.1 Yleistä tietoa

Django on Python-ohjelmointikielelle tehty ohjelmistokehys (framework), joka on suunniteltu tietokantaa käyttävien web-sovellusten nopeaan kehitykseen. Kehys tarjoaa työkaluja web-sovelluksen tietomallien ja tietokannan hallintaan, näkymien luontiin ja sivuston ulkoasun/käyttöliittymän näyttämiseen.
Django-paketin mukana tulee myös kevyt palvelinohjelmisto kehitystä auttamaan, mutta tuotantoversio sivuista laitetaan yleensä toimimaan jonkin sitä varten suunnitellun palvelimen päälle. Django toimii Pythonin versioilla 2.3 - 2.6 ja tukee PostgreSQL-, MySQL-, ORACLE- ja SQLite-tietokantoja suoraan. Lisäksi siihen on lisätty ulkopuolisten kehittäjien toimesta tuki esimerkiksi Microsoft SQL Serverille ja ODBC:lle. (Kaplan-Moss, 2007; Django-kehittäjät, 2010a; Django-kehittäjät, 2010c.)

Django sai alkunsa vuonna 2003 The World Company -lehtitalon web-osastolta, kun Adrian Holovaty ja Simon Willison alkoivat kehittää yritykselle uutta CMS-järjestelmää Pythonilla vanhan PHP-pohjaisen järjestelmän tilalle. Pythonille oli tarjolla jo tuolloin web-kehitykseen suunnattuja työkaluja, mutta mikään niistä ei täyttänyt kaikkia heidän vaatimuksiaan. Mukaan liittyi myöhemmin Jacob Kaplan-Moss. (Kaplan-Moss, 2008.)
Vuonna 2005 projektia esiteltiin PyCon 2005 -tapahtumassa ja sen saaman hyvän vastaanoton sekä Ruby on Rails -alustan kehityksen innoittamana he alkoivat jakaa CMS-projektia kahteen osaan ja kehittää sitä eteenpäin. Toisesta kehittyi kaupallinen, lehtitaloille suunnattu sisällönhallintajärjestelmä nimeltä Ellington ja toisesta yleiskäyttöinen web-sovellusten kehitykseen suunniteltu framework nimeltä Django. Djangon opensourceksi julkaisun jälkeen, se sai osakseen paljon kiinnostusta ja vapaaehtoiset kehittäjät alkoivat lähettää korjauksia ohjelmistokehyksen parantamiseksi. Djangon versio 1.0 julkaistiin loppuvuodesta 2008 ja versiota 1.2 ollaan valmistele-

massa julkaisua varten kirjoitushetkellä, keväällä 2010. (Kaplan-Moss, 2008; Django-kehittäjät, 2010c.)

### 4.2 Kehyksen toimintaperiaatteet

Django-kehys on rakennettu noudattamaan väljää liitännäisyyttä järjestelmän eri osien välillä. Vaikka eri osat työskentelevät tiivisti keskenään, ne ovat silti mahdollisimman vähän riippuvaisia toisistaan. Esimerkiksi tietokantakerros ei välitä miten sen käsittelemä data näytetään ja kehittäjä voi korvata sivujen ulkoasusta vastaavan osan kokonaan toisella ilman järjestelmän rikkoutumista. Lisäksi järjestelmän tulisi olla kaikilla tasoilla mahdollisimman yhtenäinen väljästä liitännäisyydestä huolimatta. (Django-kehittäjät, 2010c.)

Django yrittää vähentää kirjoitettavan koodin määrää ja nopeuttaa ohjelmistokehitystä silloin kun se on mahdollista. Järjestelmä huolehtii www-sovellusten kehitystä hidastavista piirteistä kehittäjän puolesta, kuten esimerkiksi automaattisen lomakkeenkäsittelyjärjestelmän käyttö tarvittaessa.

Django yrittää estää kehittäjää toistamasta itseään. Data ja toiminnot tulisi olla vain yhdessä paikassa ja järjestelmän tulisi voida järkevissä rajoissa päätellä mahdollisimman vähästä tiedosto mahdollisimman paljon. (Django-kehittäjät, 2010c.)

Lause "Explicit is better than implicit" eli karkeasti käännettynä "Yksiselitteinen toimintatapa on parempi kuin epäsuora" on yksi Python-filosofian peruspilareista ja toiminnassa myös Djangossa. Järjestelmän tulisi toimia mahdollisimman vähän "taianomaisesti" ja epäsuorasti. Selkeys ja yksinkertaisuus ovat aina parempia toimintatapoja ellei poikkeavalla tai epäselvällä toimintatavalla saavuteta jotain erityisen suurta etua. (Django-kehittäjät, 2010c; Peters, 2004.)

### 4.3 Djangon tärkeimmät osat

URL dispatcher (URL lähettäjä) on kehyksen osa, joka ohjaa käyttäjän pyynnön pyydetyn URL:n perusteella oikeaan näkymään (view). Dispatcherin toimintaa säätelee kehittäjän kirjoittama URL patterns -tiedosto, jonka avulla sivuston ja sovellusten URL-rakenteen voi suunnitella regular expression -kielellä. Regular expressionin

käyttö tarjoaa lähes rajattomat mahdollisuudet muokata sivuston osoiterakennetta. Kehittäjä voi esimerkiksi piilottaa sivuston tiedostojen tiedostopäätteet ja luoda näin selkeämpiä ja lyhyempiä osoitteita sivustolle. (Django-kehittäjät, 2010c.)

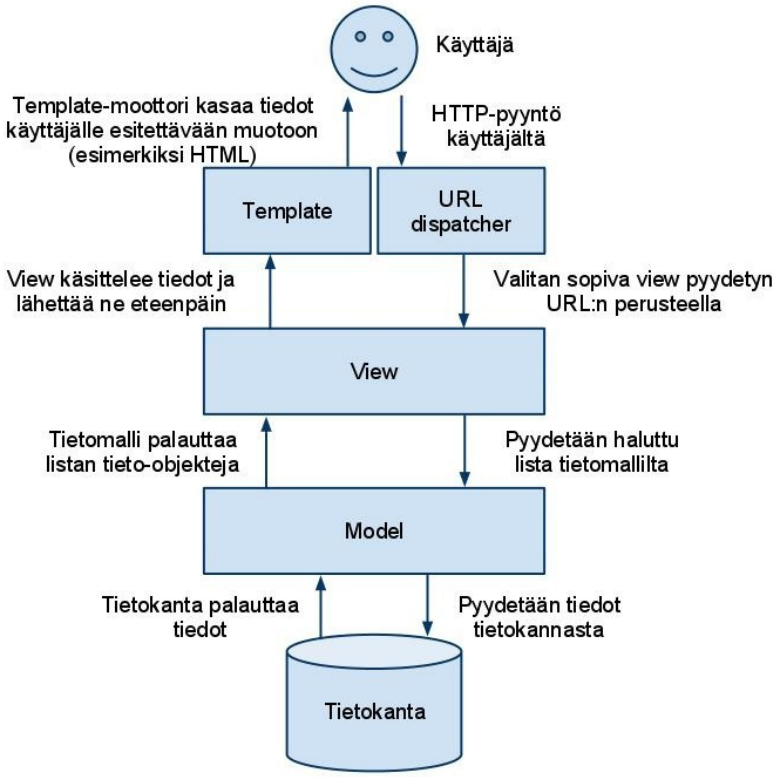
Model (tietomalli) pitää sisällään sovelluksen tietomallit. Tietomallit kuvataan suoraan Python-ohjelmointikielellä ja sen pohjalta luodaan valitun tietokantajärjestelmän tietokantaan taulut. Taulun luonnin jälkeen tietomalliobjekti toimii siltana ohjelmakoodisi ja tietokannan välillä huolehtien kyselyistä ja tietokantajärjestelmän palauttamista vastauksista. Kehittäjän ei tarvitse halutessaan koskea kertaakaan tietokanan kyselykieleen, mutta Django tarjoaa siihenkin helpottavia työkaluja tarvittaessa. (Django-kehittäjät, 2010c; Django-kehittäjät, 2010d.)

View (näkymä) hakee ja käsittelee tietomalliin kautta tietokannan tietoja ja määrittää pohjimmiltaan, mitä tietoa käyttäjälle näytetään. Se ei itsessään ota kantaa, miten tieto lopulta käyttäjälle päätyy. Näkymät toteutetaan Python-kielen funktioilla ja ottavat ainoana pakollisena parametrina järjestelmän antaman request-objektin, johon on tallennettuna käyttäjän tekemän pyynnön tiedot. Django palauttaa funktiolta saamansa sisällön käyttäjälle, mutta se muokataan yleensä parempaan ulkoasuun Djangon pohjajärjestelmän kautta. (Django-kehittäjät, 2010c; Django-kehittäjät, 2010d.)

Template (sivupohja) on kehyksen osa, joka päättää miten sille annettu tieto näytetään käyttäjälle. Template-moottori lataa levyltä ennalta kirjoitettuja pohjia, joihin sille näkymän antama tieto tai lista upotetaan ja palautetaan käyttäjälle. Useimmiten template-moottori käsittelee HTML- tai XHTML-tiedostoja, mutta sillä voidaan tuottaa mitä tahansa tekstimuotoista tiedostoformaattia. Sivupohjajärjestelmä on suunniteltu www-sivujen suunnittelijoiden käyttöön, mutta sisältää jonkin verran ohjelmointia mahdollistavia piirteitä. Turvallisuuden ja selkeyden takia ohjelmointiominaisuudet on tehty mahdollisimman yksikertaisiksi. (Django-kehittäjät, 2010c.)

Yksi template-moottorin hyödyllisistä piirteistä on perintäjärjestelmä, joka mahdollistaa sivuston ulkoasun rakentamisen modulaarisesti. Kehittäjä voi esimerkiksi tehdä koko sivuston määritykset kattavan pohjan, jonka yksittäisen sivun määritykset sisältävä pohja perii. Perintä tehostaa ja selkeyttää sivuston ulkoasun kehitystä ja ylläpitoa, mutta vaatii opettelua ennen tuotantokäyttöä.

Kuvassa 1 on havainnollistettuna käyttäjän lähettämän HTTP-pyynnön eteneminen www-palvelimella ja Django-kehyksessä.




Kuva 1. Django-ohjelmistokehyksen tärkeimmät osat ja käyttäjän pyynnön eteneminen

Django tarjoaa valmiina automaattisen hallintapaneelin, joka luodaan suoraan kehittäjän määrittelemien tietomallien pohjalta. Se tarjoaa järjesteltävän listan objekteista, syöttö- ja muokkauslomakkeen niille ja objektien poistamisen. Hallintapaneelia voidaan muokata sivuston tai käyttäjien tarpeisiin sopivaksi ja se on suunniteltu sellaisten henkilöiden käyttöön, joilla ei ole teknistä asiantuntemusta. Tarjolla on myös valmiit työkalut kirjautumisjärjestelmän toteuttamiseen ja sivuston toiminnan optimointiin erilaisten välimuistijärjestelmien avulla. (Django-kehittäjät, 2010c.)

## 4.4 Lisäosat

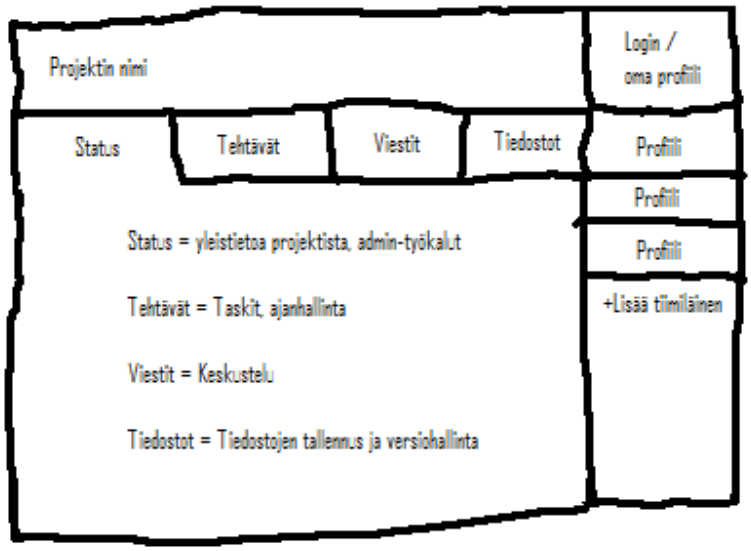
Djangon filosofia suosii www-sovelluksen rakentamista modulaarisesti. Tämän johdosta lähes jokaisesta projektikohtaisesta Django-sovelluksesta voi muokata helposti yleiskäyttöisen moduulin, joka voidaan asentaa nopeasti toisiin projekteihin. Ideaalitilanteessa yksi sovellus tekee vain yhden asian todella hyvin ja kehittäjälle jää tehtäväksi vain sitoa sovellukset yhteen ja luoda järjestelmälle yhtenäinen käyttöliittymä. Osa Django-kehittäjistä on paketoinut luomansa yleiskäyttöiset sovellukset ja jakavat niistä muiden kehittäjien käytettäväksi. (Ölçgen, 2010.)
Uudelleenkäytettävien sovellusten idean ympärillä on rakennettu muutama sivusto, jolla näitä sovelluksia listataan. Eräs näistä sivustoista on Djangozen.com <URL:http://djangozen.com/plugins/> Sovellusten sivuilla on yleensä kehittäjän yhteystiedot ja linkki sovelluksen kotisivuille. Osa uudelleenkäytettävistä sovelluksista on päässyt mukaan joidenkin Linux-jakelujen pakettijärjestelmiin, josta ne voidaan asentaa ja päivittää helposti. Myös Djangon omassa wikissä on listattuna muutamia sovelluksia. Sovelluksen paketin latauksen jälkeen sen sisältö puretaan Django-projektin alle, lisätään sovelluksen hakemiston nimi asetustiedostoon, lisätään URL patterns -tiedostoon ohjauskäsky ja käsketään Djangoa luomaan sovelluksen haluamat tietokantataulut. Tämän jälkeen sovellus on valmis käytettävissä omalla sivustolla. (Django-kehittäjät, 2010c; Django-kehittäjät, 2010d.)

## 4.5 Käyttö

Djangoa käytetään sen kehittäneen The World Companyn www-sivuilla, kuten paikallisia tapahtumia ja uutisia listaavassa Lawrence.com:issa ja yleisemmällä uutissivustolla LJWorld.com:issa. Myös Washington Post on käyttänyt Djangoa muutamien projektiensa toteuttamiseen. (Django-kehittäjät, 2010e.)
Djangoa käyttävien sivustojen listaukseen on luotu myös Djangosites.org -sivusto <URL:http://www.djangosites.org/>, jonne kehittäjät voivat rekisteröidä sivustonsa ja saada niistä kommentteja sekä arvioita. Kirjoitushetkellä järjestelmässä on 3174 sivustoa.

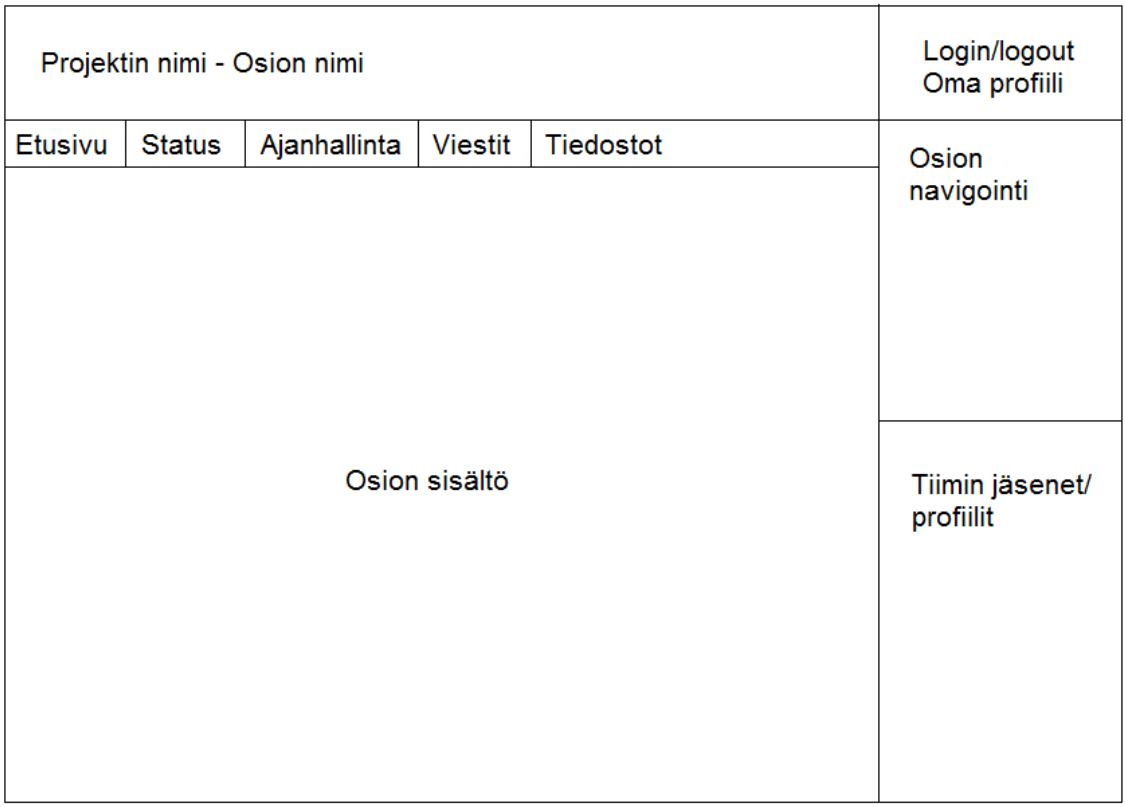
# 5 OMA TOTEUTUS

## 5.1 Työn suunnittelu




Kuva 2. Ensimmäinen tekemäni käyttöliittymäsuunnitelma.

Opinnäytetyökseni suunnittelin ja toteutin web-pohjaisen projektinhallintajärjestelmän Python-ohjelmointikielellä ja Django-ohjelmistokehyksellä. Järjestelmä on tarkoitettu pienille ryhmille, jotka tarvitsevat kevyttä ja helppokäyttöistä työkalua projektien hallintaan. Järjestelmän on tarkoitus tukea useita käyttäjiä ja projekteja, joita käyttäjät voivat luoda ja joihin heitä voidaan osoittaa. Idea järjestelmän toteutukseen lähti omista tarpeista, koska olisi useaan otteeseen tarvinnut koulutukseni ohessa projektityökaluja muutaman viikon projekteihin. Tarjolla olleet työkaluvaihtoehdot olivat joko liian monimutkaisia tai liian isoja pieniin projekteihini. Suunnittelin sivustolle seuraavat suuremmat osiot: projektin yleistiedot, ajanseurantajärjestelmä, viestijärjestelmä ja tiedostonhallintajärjestelmä.

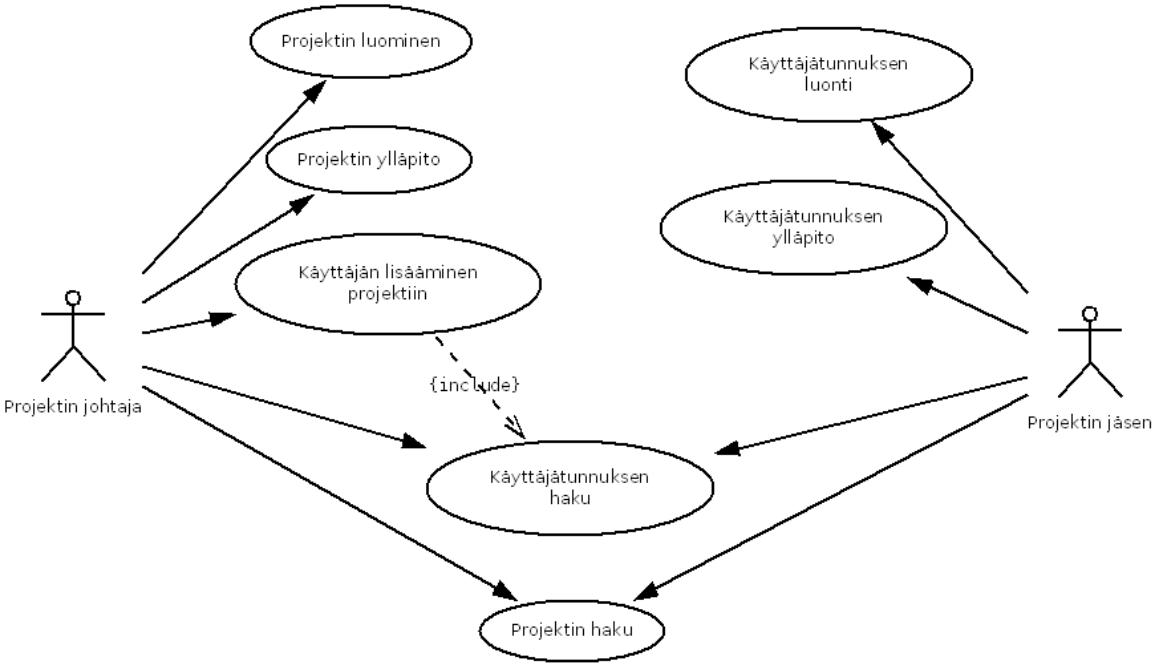



Kuva 3. Suunnitelmien pohjalta muokkaamani käyttöliittymäsuunnitelma.

### 5.1.1 Projektin yleistiedot

Projektin yleistietoihin on tarkoitus tulla projektin tietojen lisäksi siihen osoitetut jäsenet. Jäsenlista saadaan sivustolle rekisteröityjen käyttäjien listasta. Lisäksi sivuilla tulisi näkyä yhteenveto projektin tärkeistä luvuista ja tiedoista kuten esimerkiksi projektiin käytetty työaika, uusimmat tiedostot, vähiten valmistuneet tehtävät ja lista projektin jäsenistä.

Projekti-tietomallin mahdolliset kentät on projektin nimi, luontipäivämäärä ja -aika, kuvaus projektista, lista projektin jäsenistä, projektin johtaja/omistaja ja asetuksia projektin käyttöoikeuksista.



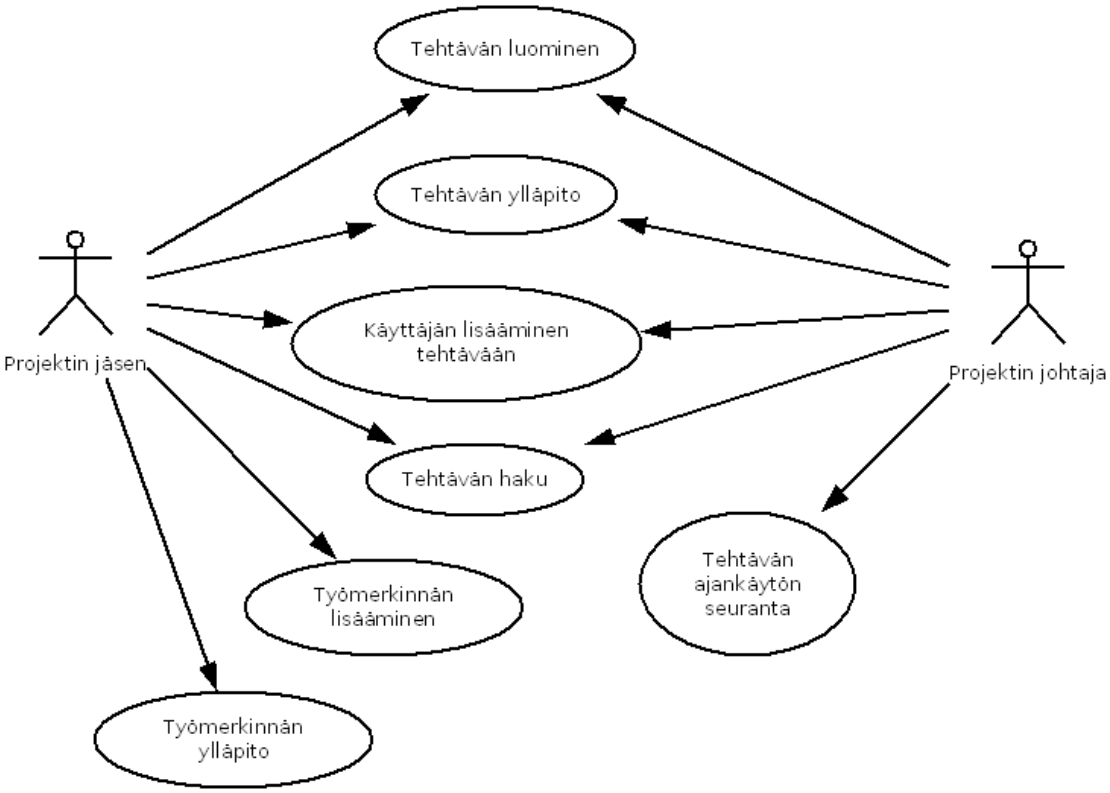

Kuva 4. Projektin ja käyttäjän käyttötapauskaavio.

### 5.1.2 Ajanhallinta

Ajanhallintaosion tarkoitus on pitää kirjaa projektin tehtävistä, niiden tiedoista ja niihin osoitetuista projektin jäsenistä sekä tehtäviin tehdyn työn ylläpidosta. Tehtävien toteutuneet tunnit lasketaan työmerkintöjen pohjalta.

Tehtävä-tietomallin mahdolliset kentät on projekti, johon tehtävä kuuluu, luontipäivämäärä ja -aika, tehtävän nimi, kuvaus tehtävästä, aloituspäivämäärä, deadline-päivämäärä, suunnitellut tunnit, valmistumisprosentti ja tehtävään osoitetut projektin jäsenet.

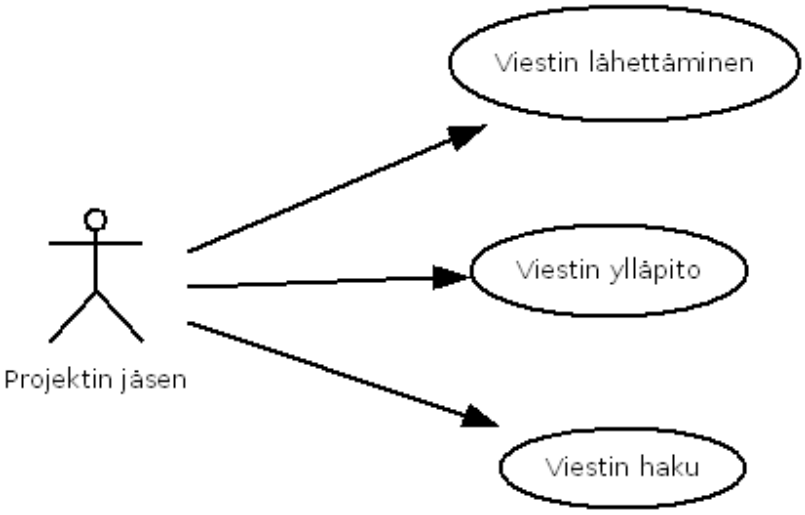
Työmerkintä-tietomallin mahdolliset kentät on tehtävä, johon työmerkintä kuuluu, luontipäivämäärä ja -aika, kuvaus tehdystä työstä, aloituspäivämäärä ja -aika, lopetuspäivämäärä ja -aika ja työn tehnyt projektin jäsen.




Kuva 5. Tehtävän ja työmerkinnän käyttötapauskaavio.

### 5.1.3 Viestijärjestelmä

Viestiosion tarkoitus on tarjota projektin jäsenille mahdollisuus lähettää projektiin liittyviä viestejä. Viestit esitetään aikajärjestyksessä vanhin ensin ja viestimäärän kasvaessa viestilista pilkotaan useammalle sivulle lukemisen helpottamiseksi. Viesti-tietomallin mahdollisia kenttiä on projekti, johon viesti kuuluu, viestin sisältö, lähetyspäivämäärä ja -aika ja viestin lähettänyt projektin jäsen.

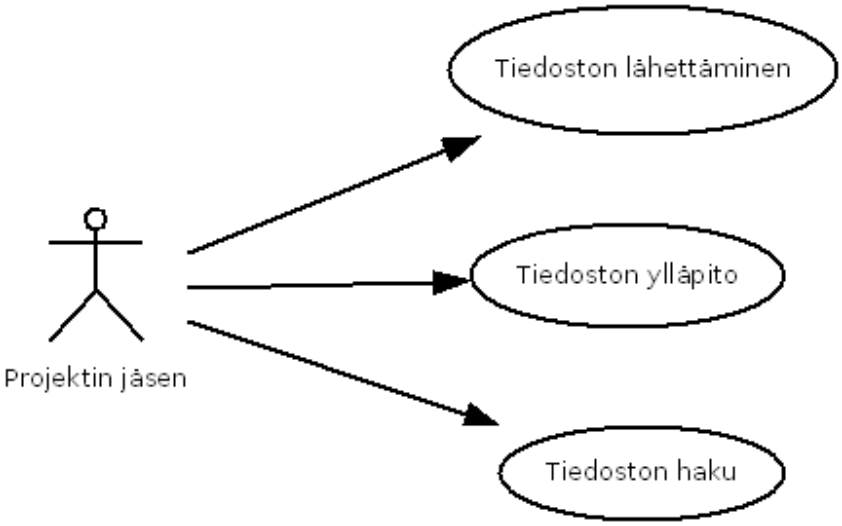



Kuva 6. Viestin käyttötapauskaavio.

### 5.1.4 Tiedostonhallinta

Tiedostohallintaosion tarkoitus on tarjota projektin jäsenille paikka tallettaa ja jakaa projektiin liittyviä tiedostoja. Tiedostoille on tarjolla mahdollisuus versiohallintaan, jossa samannimisistä tiedostoista tehdään palvelimelle ladatessa uusi versio edellisen tilalle säilyttäen vanhat versiot.

Tiedosto-tietomallin mahdollisia kenttiä on projekti, johon tiedosto kuuluu, tiedoston nimi, lähettäjän kommentti, itse tiedosto, lähetyspäivämäärä ja -aika ja tiedoston lähettänyt projektin jäsen.



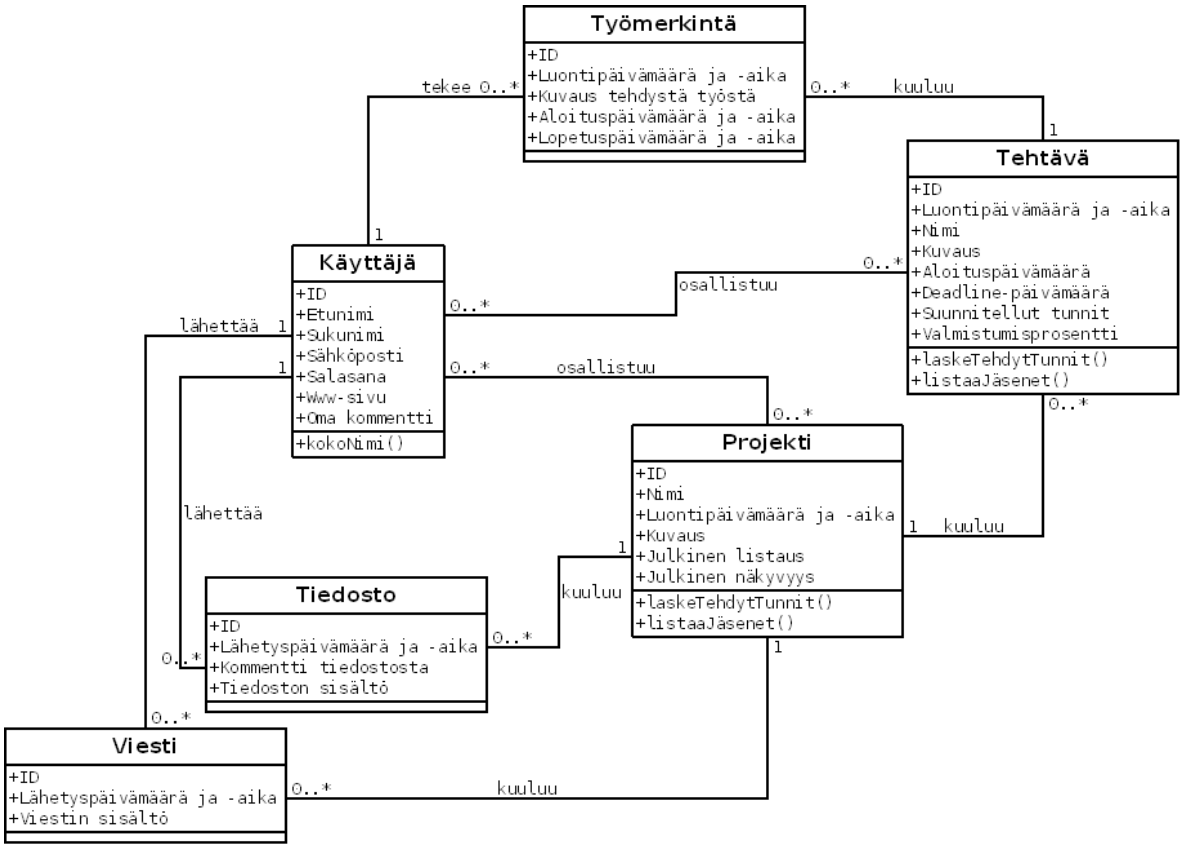
Kuva 7. Tiedoston käyttötapauskaavio.

### 5.1.5 Käyttäjät

Sivuston käyttäjille on tarjolla mahdollisuus rekisteröityä järjestelmään. Jokaisella projektijärjestelmään rekisteröityneellä käyttäjällä on oma sivu, jolle he voivat laittaa itseään koskevia tietoja ja lista kaikista projekteista, jossa hän on mukana. Käyttäjä-tietomallin mahdollisia kenttiä on etunimi, sukunimi, sähköposti, salasana, käyttäjän www-sivu ja kommentti.

### 5.1.6 Luokkakaavio

Kuvassa 8 on suunniteltujen tietomallien perusteella piirtämäni luokkakaavio.

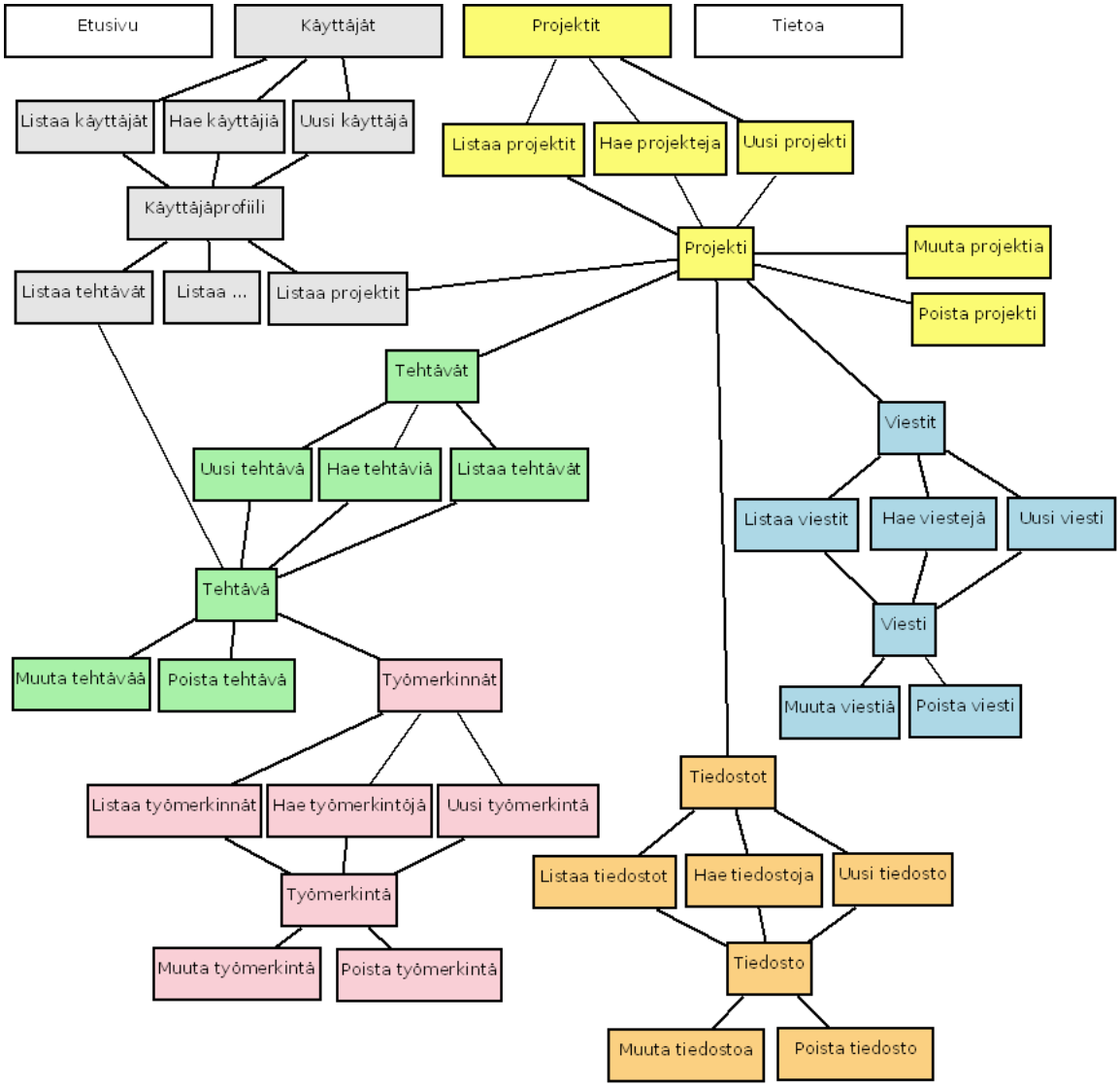



Kuva 8. Järjestelmän luokkakaavio.

### 5.1.7 Käyttöliittymä

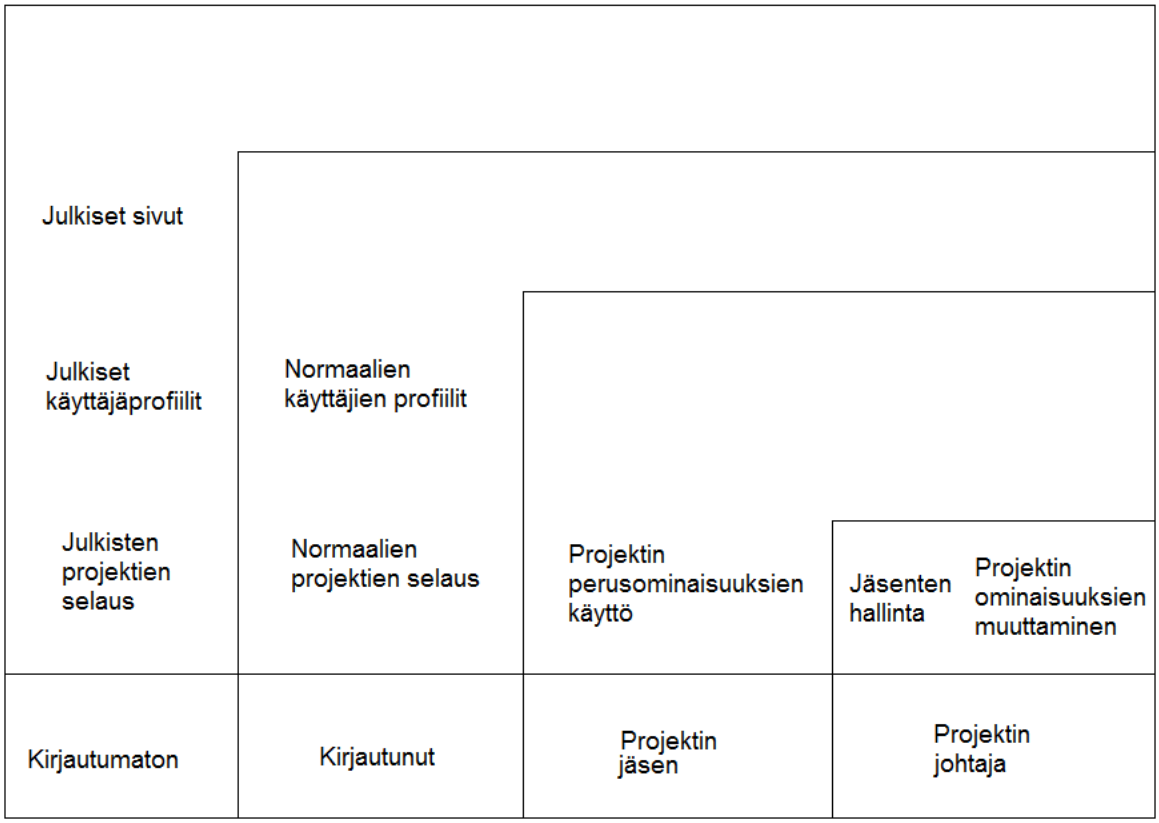
Järjestelmän käyttöliittymäksi luodaan www-sivusto, jossa on mahdollisuus projektin tietojen listaukseen, hakuun ja ylläpitoon. Sivuston ulkoasun tulee olla selkeä ja kevyt. Suurin osa projektin tallennetuista tiedoista on tekstiä, joten luettavuuteen ja tekstin jaotteluun on kiinnitettävä huomiota.

Järjestelmään tallennetaan paljon erilaista tietoa ja sen tietomalleilla on muutamia tärkeitä toimintoja, joten sivuston eri osioiden tulee olla loogisesti ja selkeästi jaoteltuja. Kuvassa 9 on suunnitelmieni pohjalta tehty kaavio sivuston navigointirakenteesta.




Kuva 9. Sivustolle suunnittelemani navigointikaavio.

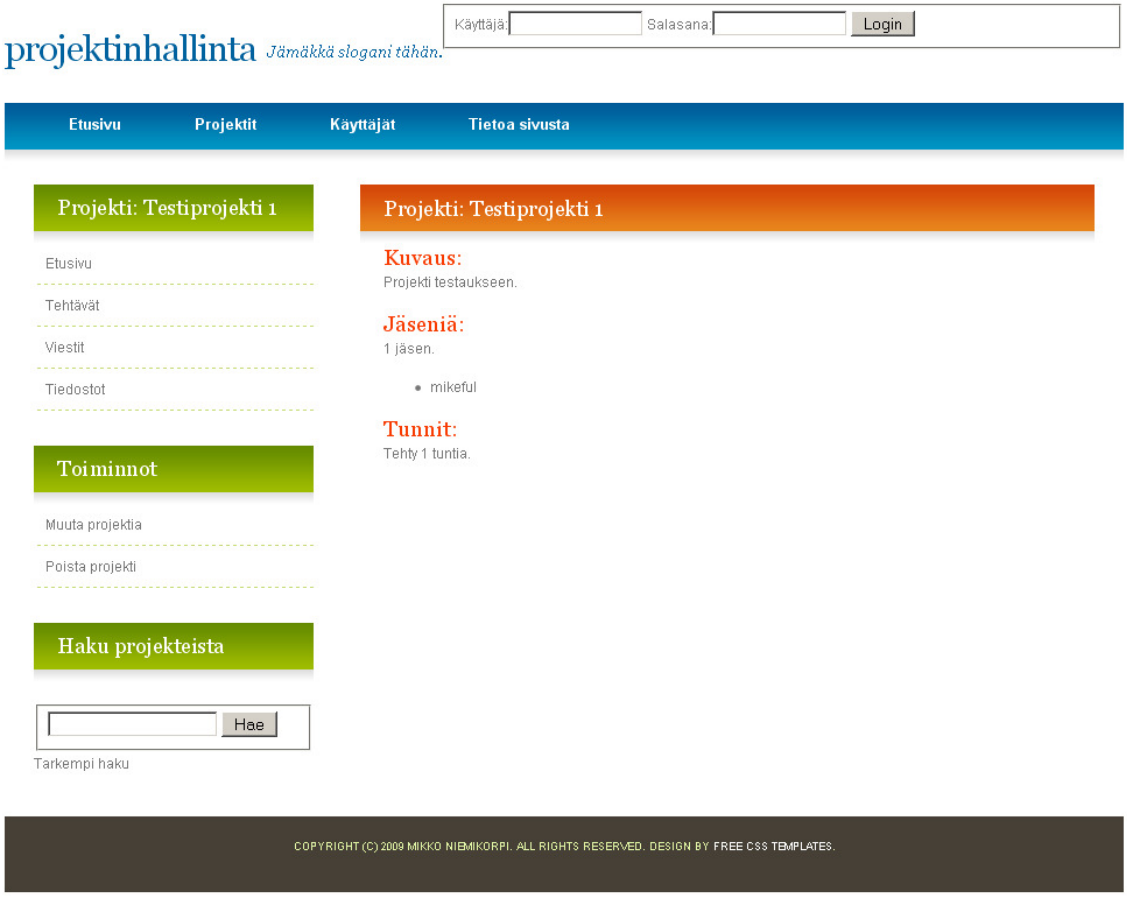
Projekteihin saatetaan tallentaa hyvinkin tärkeää tai salaista tietoa, joten sivuston tietoturvasta pitää huolehtia asiallisesti. Erilaisille käyttäjille pitää luoda erilaiset käyttöoikeudet sivuston osiin ja kirjautumattomille projekteja ja niiden tietoja ei näytetä lainkaan ilman erillistä asetusta. Kuvassa 10 on suunnittelemani käyttöoikeustasot ja niihin liittyvät oikeudet. Sisemmän tason käyttäjällä on kaikki ulompien tasojen oikeudet.




Kuva 10. Sivustolle suunnittelemani turvatasot.

## 5.2 Työn toteutus

Toteutin työn Python-ohjelmointikielellä ja Django-kehyksellä. Suunnittelemani luokat luotiin Djangon tarjoaman alisovellusjärjestelmän avulla, jossa toteutin luokat omissa sovelluksissa. Sovellukset toimivat itsenäisesti ja yhdessä sovelluksessa voi olla useita tietomalleja ja näkymiä niille. Selitän tässä luvussa tarkemmin toteuttamiani sovelluksia.




Kuva 11. Kuvankaappaus toteuttamastani sivustosta.

### 5.2.1 Projektin yleistiedot

Projektin yleistiedot on toteutettu järjestelmässä Projektit-nimisessä sovelluksessa, jossa on Projekti-tietomalli. Tietomallissa on kentät projektitunnukselle, projektin nimelle ja kuvaukselle, luontipäivämäärälle, asetukset projektin julkisen katselun ja listauksen sallimiseen ja lista projektin jäsenistä. Jäsenlistaan voi valita henkilöitä Djangon tarjoaman käyttäjäjärjestelmän tietomallista.

Projektit-sovelluksessa on näkymät projektien listaukselle, yhden projektin näyttämiselle, projektin luomiselle ja muokkaamiselle, projektin poistamiselle ja projektien haulle. Hakunäkymä hakee käyttäjän hakusanaa projektien nimistä ja kuvauskentistä.

### 5.2.2 Ajanhallinta

Ajanhallintaa varten järjestelmässä on Aika-niminen sovellus, jossa on Tehtävä-tietomalli. Tietomallissa on kentät tehtävätunnukselle, projektille, johon tehtävä kuuluu, tehtävän nimelle ja kuvaukselle, luontipäivämäärälle, aloitus- ja deadlinepäivämäärille, suunnitelluille tunneille, tehdyille tunneille ja lista tehtävään määrätyistä projektin jäsenistä.

Aika-sovelluksessa on näkymät tehtävien listaukselle, yhden tehtävän näyttämiselle, tehtävän luomiselle ja muokkaamiselle, tehtävän poistamiselle ja tehtävien haulle. Hakunäkymä hakee käyttäjän hakusanaa tehtävien nimistä ja kuvauskentistä.

Tehtävien ajanseurantaa varten tehty Työmerkintä-tietomalli oli Aika-sovelluksessa mukana, mutta sen kanssa tuli ongelmia joiden ratkaisu olisi vaatinut lähes koko järjestelmän navigointipuolen uudelleenkirjoituksen. Jätin työmerkinnät pois ja korvasin sen yksinkertaisemmalla, mutta epätarkemmalla aikaseurannalla "tehdyt tunnit"-kentällä Tehtävä-tietomallissa. Työmerkinnälle on olemassa toimivat näkymät.

### 5.2.3 Viesti- ja tiedostojärjestelmä

Viestijärjestelmä on toteutettu järjestelmässä Viestit-nimisenä sovelluksena, jossa on Viesti-tietomalli. Tietomallissa on kentät viestitunnukselle, projektille, johon viesti kuuluu, viestin sisällölle, luontipäivämäärälle ja -ajalle ja viestin lähettäneelle projektin jäsenelle.

Viestit-sovelluksessa on näkymät viestien listaukselle, yhden viestin näyttämiselle, viestin lähettämiselle ja muokkaamiselle, viestin poistamiselle ja viestien haulle. Hakunäkymä hakee käyttäjän hakusanaa viestien sisällöistä.

Tiedostonhallinta on toteutettu järjestelmässä Tiedostot-nimisenä sovelluksena, jossa on Tiedosto-tietomalli. Tiedostonhallinta on lähes tietomalliltaan ja näkymiltään lähes sama kuin viestijärjestelmä. Lisäsin viestijärjestelmään verrattuna tiedoston nimi -kentän, muutin viestin sisältö -kentän tiedoston kommenttikentäksi ja lisäsin kentän itse tiedostoa varten. Näkymissä muutin tiedostolistauksien kentät uusille nimille, muutin haun hakemaan tiedoston nimestä ja kommentista sekä muutin lähetyslomakkeen käsittelemään lisätyn tiedostokentän.

### 5.2.4 Käyttäjät

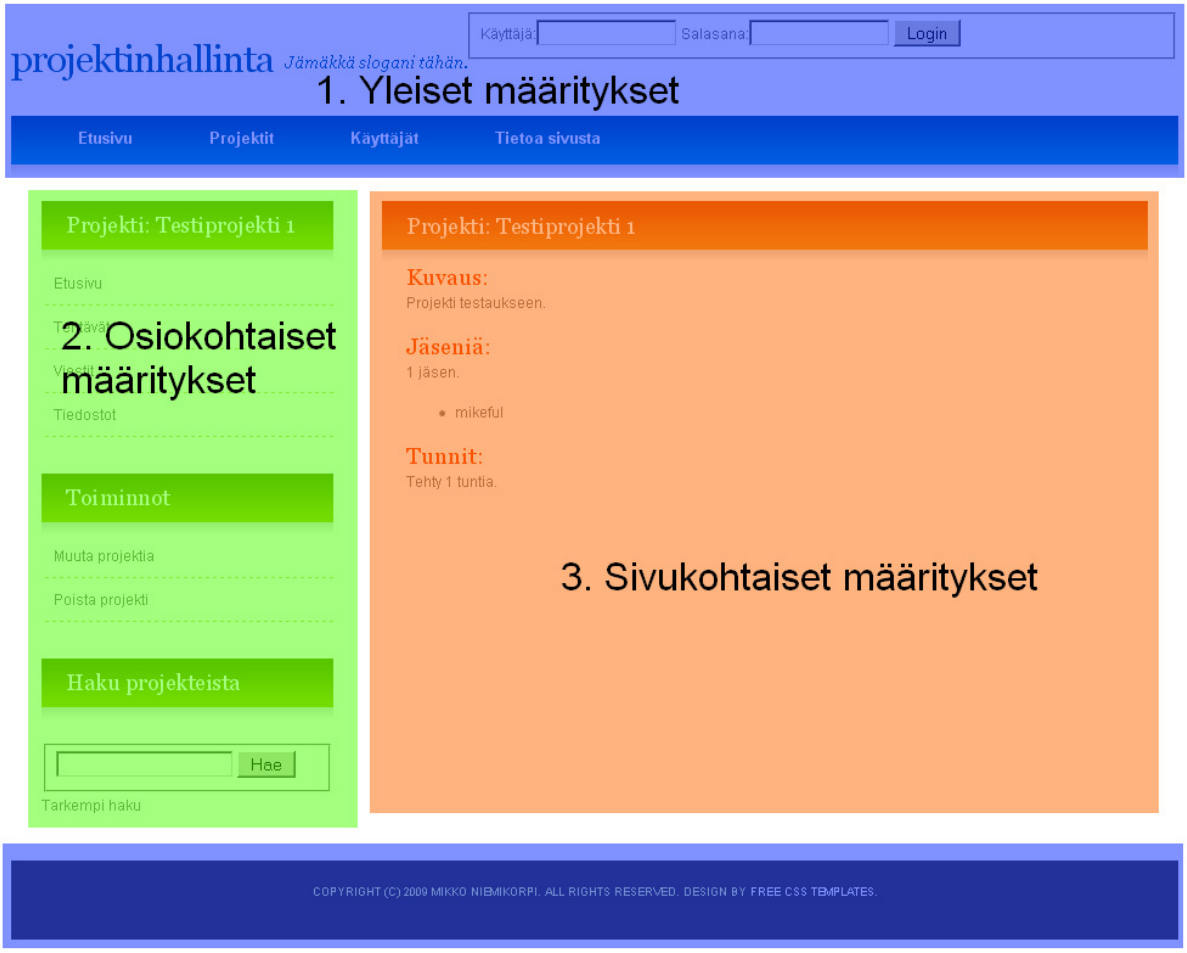
Django tarjoaa valmiin käyttäjäjärjestelmän, joten käytin sitä ja laajensin tarjottua tietomallia lisäämällä siihen käyttäjän www-sivulle ja kommentille kentät. Muissa sivuston tietomalleissa oli helppo käyttää Käyttäjä-tietomallia ja sain sillä projektin työntekijät helposti ilmoitettua.

Käyttäjien rekisteröityminen ja profiilisivut puuttuvat. Löysin kumpaakin varten toisten kehittäjien tekemät Django-sovellukset, mutta en ehtinyt laittaa niitä toimintaan. Näiden käyttöönoton jälkeen olisin voinut luoda kirjautumisen järjestelmään ja toteuttaa turvallisuustasot järjestelmään.

### 5.2.5 Ulkoasu

Ulkoasun pohjana käytin FreeCSSTemplates.org-sivuston tarjoamaa ilmaista TriColor-nimistä sivupohjaa, joka ladattavissa osoitteesta <URL:http://www.freecsstemplates.org/preview/tricolor/>. Sivupohjan oikeassa yläreunassa oli paikka hakulaatikolle, mutta korvasin sen kirjautumislomakkeella ja siirsin osiokohtaisen hakulomakkeen sivupohjan vasemmassa laidassa olevaan sivupalkkiin.

Muokkasin sivupohjan Djangon käyttämälle sivupohjajärjestelmälle kolmella tasolla toimivaksi pohjaksi. Ensimmäisellä tasolla on koko sivuston käyttämät tyylimäärittelyt ja sivujen rakenteet. Toisella tasolla on sivuston päätasojen omat sisältömääritykset ja ulkoasu. Kolmannella tasolla on yksittäisen sivun sisältömääritykset. Kuvassa 12 on havainnollistettu eri tasojen vaikutusta sivun ulkoasuun.




Kuva 12. Pohjajärjestelmän tasot, joilta sivun ulkoasu saa määritykset

## 5.3 Jatkokehitys

Projektinhallintajärjestelmää toteuttaessani opin sekä Pythonista että Djangosta uusia piirteitä, joiden takia haluaisin aloittaa koko projektin alusta ja hyödyntää saamaani kokemusta ja osaamista.

Järjestelmään pitäisi lisätä vielä käyttäjien rekisteröityminen ja kirjautuminen, käyttäjien profiilisivut sekä käyttäjien projektikohtaiset oikeudet. Osan näistä voisi toteuttaa olemassa olevilla "django-profiles"- ja "django-registeration"-lisäosilla. Nykyinen järjestelmä ei noudata omasta mielestäni riittävästi Djangon suosimaa modulaarista järjestelmäsuunnittelua. Uudelleenkirjoituksen yhteydessä olisi kätevää samalla pilk-

koa projektia helpommin käsiteltäviin osiin ja samalla uskoisin uusien alisovellusten toteuttamisen helpottuvan. Aiheesta löytyy jonkin verran artikkeleita ja tietomalleja ei tarvitsisi muuttaa kovinkaan paljoa.

En hyödyntänyt Djangon tarjoamia testaustyökaluja riittävästi. Tarjolla on järjestelmä, jossa näkymille voi syöttää testidatana ohjelmallisesti luotuja pyyntöjä parametreina ja tarkistaa näkymän palauttaman syöte. En myöskään testannut sivujen toimivuutta ja käytettävyyttä muilla selaimilla kuin Mozilla Firefoxilla ja Google Chromella enkä mobiililaitteiden selaimilla. Djangoon on tarjolla sovelluksia, jotka vaihtavat sivuston template-moottorin käyttämiä sivupohjia selaimen tai päätelaitteen mukaan.

# 6 YHTEENVETO

Asetin itselleni tavoitteeksi oman projektityökalun suunnittelemisen ja toteuttamisen Python-kielellä ja Django-kehyksellä sekä käytännön kokemuksen hankkiminen valitsemistani työkaluista. Kummatkin tavoitteet ovat omasta mielestäni täyttyneet hyvin. Sain toteutettua projektityökalun ja opin samalla uusia asioita sekä yleisesti sovelluskehityksestä että erityisesti Pythonin ja Djangon yhdistelmästä.
Opinnäytetyöstäni saamalla kokemuksella aloittaisin vastaavanlaisen projektin toteuttamisen hiukan eri tavalla. Oman työn aikataulutus ei sujunut aivan kuten olisin halunnut ja tulevissa projekteissa ajattelin asettaa itselleni tiukemmat välitavoitteet.
Koulussa opiskelluille asioille oli opinnäytetyön edetessä jatkuvasti tarvetta ja ilman käymääni koulutusta en olisi todennäköisesti selvinnyt projektistani. Python-osaamiseni oli jo ennen projektin alkua kohtuullisen hyvällä tasolla enkä tunnu oppineeni mitään erityisen uutta itse kielestä. Sen sijaan Djangosta saamani kokemuksen uskon osoittautuvat hyödylliseksi tulevaisuudessa ja ajattelin opiskella kehystä lisää.
Djangosta voisi tulla myös jollekin muullekin hyvä aihe opinnäytetyöhön, jos Python-kieli on jo tuttu kieli ja haluaisi tutustua sen tarjoamiin työkaluihin enemmän.

## LÄHTEET

Django-kehittäjät 2010. Django documentation, Databases. [online]. [viitattu 21.4.2010]. Saatavilla www-muodossa: <URL:http://docs.djangoproject.com/en/1.1/ref/databases/>

Django-kehittäjät 2010. Django documentation, Design philosophies. [online]. [viitattu 21.4.2010]. Saatavilla www-muodossa: <URL:http://docs.djangoproject.com/en/dev/misc/design-philosophies/>

Django-kehittäjät 2010. Django documentation, Django at a glance. [online]. [viitattu 19.4.2010]. Saatavilla www-muodossa: <URL:http://docs.djangoproject.com/en/dev/intro/overview/>

Django-kehittäjät 2010. Django documentation, FAQ: General. [online]. [viitattu 21.4.2010]. Saatavilla www-muodossa: <URL:http://docs.djangoproject.com/en/1.1/faq/general/>

Django-kehittäjät 2010. Django kotisivut. [online]. [viitattu 26.4.2010]. Saatavilla www-muodossa: <URL:http://www.djangoproject.com/>

Django-kehittäjät 2010. Django resources, Django application components. [online]. [viitattu 26.4.2010]. Saatavilla www-muodossa: <URL:http://code.djangoproject.com/wiki/DjangoResources#Djangoapplicationcomponents>

dotProject-yhteisö 2006. dotProject-wiki, Main Page. [online]. Päivitetty joulukuussa 2007. [viitattu 20.4.2010]. Saatavilla www-muodossa: <URL:http://docs.dotproject.net/index.php?title=Main_Page>

dotProject-yhteisö 2006. dotProject-wiki, System Prerequisites. [online]. Päivitetty huhtikuussa 2010. [viitattu 20.4.2010]. Saatavilla www-muodossa: <URL:http://docs.dotproject.net/index.php?title=System_Prerequisites>

dotProject-yhteisö 2006. dotProject-wiki, What is Included. [online]. Päivitetty helmikuussa 2007. [viitattu 20.4.2010]. Saatavilla www-muodossa: <URL:http://docs.dotproject.net/index.php?title=Module_Descriptions>

Harrison, Matt 2006. Python at Google (Greg Stein - SDForum). [online]. [viitattu 26.4.2010]. Saatavilla www-muodossa: <URL:http://panela.blog-

city.com/python_at_google_greg_stein__sdforum.htm>
Ilkka Haikala & Jukka Märijärvi 2006. Ohjelmistotuotanto. Helsinki. Talentum Media Oy.
Kaplan-Moss, Jacob 2007. Django: Web Development for Perfectionists with Deadlines. Videoitu esitys. [online]. [viitattu 19.4.2010]. Saatavilla www-muodossa: <URL:http://www.youtube.com/watch?v=p-WXiqrzAf8>
Kaplan-Moss, Jacob 2008. Django: The First Five Years. Videoitu esitys. [online]. [viitattu 21.4.2010]. Saatavilla www-muodossa: <URL:http://www.youtube.com/watch?v=aJ_7KtjIVLQ>
Lutz, Mark 2006. Programming Python, Third Edition. Sebastopol, CA. O'Reilly Media Inc.
Microsoft 2006. Microsoft Office Project Standard 2007 product guide. [online]. [viitattu 24.4.2010] Saatavilla www-muodossa: <URL:http://office.microsoft.com/en-gb/project/HA101680121033.aspx>
Microsoft 2010. Microsoft Office Project Standard 2007 product overview. [online]. [viitattu 24.4.2010] Saatavilla www-muodossa: <URL:http://office.microsoft.com/en-gb/project/HA101656381033.aspx>
Microsoft 2010. Microsoft Office Project Standard 2007 version comparison. [online]. [viitattu 24.4.2010] Saatavilla www-muodossa: <URL:http://office.microsoft.com/en-gb/project/FX101759351033.aspx>
mozrat 2009. django-project-management. [online]. [viitattu 26.4.2010]. Saatavilla www-muodossa: <URL:http://code.google.com/p/django-project-management/>
Peters, Tim 2004. PEP 20 -- The Zen of Python. [online]. Päivitetty lokakuussa 204. [viitattu 26.4.2010]. Saatavilla www-muodossa: <URL:http://www.python.org/dev/peps/pep-0020/>
Python Software Foundation 2002. Mission Statement of the Python Software Foundation. [online]. Päivitetty syyskuussa 2009. [viitattu 19.4.2010]. Saatavilla www-muodossa: <URL:http://www.python.org/psf/mission/>
Redmine-yhteisö 2007. Redmine projects, Redmine. [online]. [viitattu 20.4.2010]. Saatavilla www-muodossa: <URL:http://www.redmine.org/projects/redmine>

Redmine-yhteisö 2007. Redmine-wiki, Overview. [online]. Päivitetty maaliskuussa 2010. [viitattu 20.4.2010]. Saatavilla www-muodossa: <URL:http://www.redmine.org/projects/redmine/wiki>

Redmine-yhteisö 2008. Redmine-wiki, Join us and say: "We are using Redmine!". [online]. Päivitetty maaliskuussa 2010. [viitattu 20.4.2010]. Saatavilla www-muodossa: <URL:http://www.redmine.org/wiki/redmine/TheyAreUsingRedmine>

Van Rossum, Guido 2008. Guido van Rossum - Personal Home Page. [online]. [viitattu 26.4.2010]. Saatavilla www-muodossa: <URL:http://www.python.org/~guido/>

Ölçgen, Atamert 2010. Developing Reusable Django Apps. [online]. [viitattu 26.4.2010]. Saatavilla www-muodossa: <URL:http://www.muhuk.com/2010/01/developing-reusable-django-apps/>